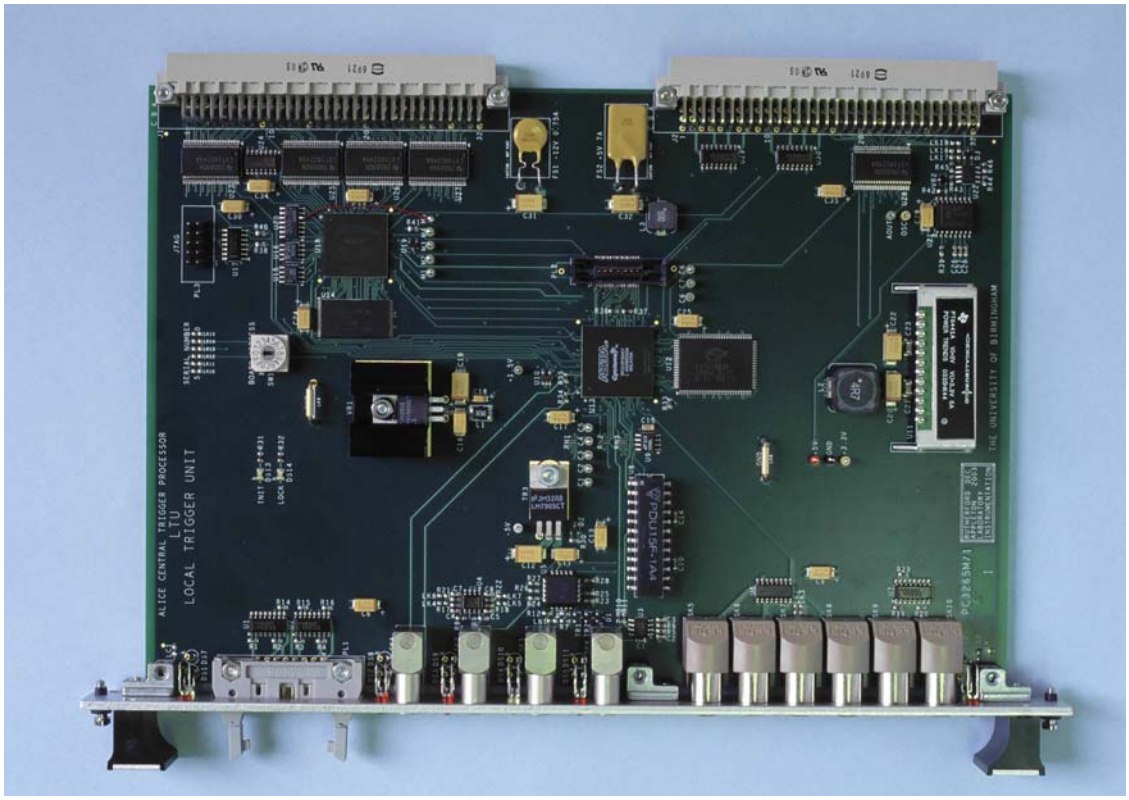
ALICE LOCAL TRIGGER UNIT

USER'S GUIDE



ALICE Trigger Group
Birmingham August 2004

1. Introduction.

The Local Trigger Unit (LTU) serves both as an interface between the detector and the Central Trigger Processor (CTP), and as an *emulator* for the CTP when the LTU is running in standalone mode. The functions of the LTU are described in the *Local Trigger Unit Preliminary Design Report*[1]. The purpose of this document is to describe how to use the LTU, and to illustrate some of the procedures available for it.

2. Getting Started.

In order to run the LTU, it is necessary first to load the software required to operate it. Installation procedures are described in Appendix A. The default platform for this system is LINUX Redhat 7.3.

To start the application type, from the processor to which the LTU is connected, type

```
vmecrate ltu
```

This will open a window with a panel representing various LTU functions, as shown in figure 1. Clicking any one of them opens further windows giving more options.

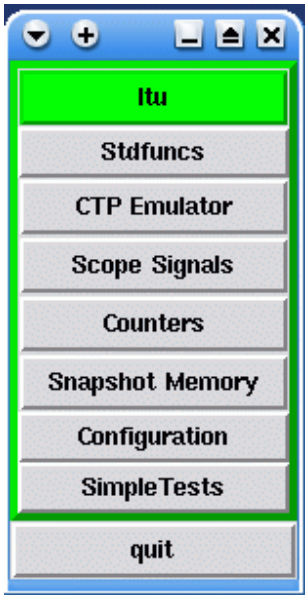


Figure 1. *Main LTU window.Each of the buttons opens a further window, as described below.*

[1] P. Jovanovic et al. Local Trigger Unit Preliminary Design Report ALICE/nnnn Birmingham 2002.

3. Main Menu

3.1 ltu

Opens executable allowing command line interface to the LTU. The window also serves to display some log messages. The window will also be opened *automatically* if the messages are needed for another process. Direct use of the command line interface is only recommended for expert users.

3.2 Stdfuncs

This gives a choice of two windows: **VME r/w** and **multiregs**.

- **VME r/w.** This allows the user to read and/or write values from/to specified VME addresses, once per use of the window. The details on how this window can be used are given in section 4.1.

- **multiregs.** This allows combined readings of more than one counter, either as a single shot or periodically. Details are given in section 4.2.

3.3 CTP Emulator

This opens a new window called **CTP Emulator** giving access to the emulation functions. From this window functions such as emulation sequence choice and editing, the definition of errors allowed in the simulation and their frequency, and so forth, can be programmed. Details are given in section 5.

3.4 Scope Signals

This allows a choice of two signals which can be allocated to the scope front panel outputs A and B. The signals can be chosen from a drop-down list with a tear-off facility. See section 6 for further details

3.5 Counters

VME counters and timers can be selected and read or cleared, either manually or periodically. See section 7 for further details.

3.6 Snapshot Memory

Opens **Snapshot Memory** window. Snapshots record the selected signals for every bunch crossing for a period of 20 ms. This period may be chosen to *start* or *stop* at a given signal. More details are given in section 8.

3.7 Configuration

This allows the user to set the principal LTU parameters, *e.g.* whether it should run in a global or standalone mode, which BUSY inputs should be used, and the details of the TTC interface. There is also an **ADC scan** function, allowing synchronization of signals within one B.C. See section 9 for further details.

3.8 Simple Tests

This allows the user to execute a few simple board tests, *e.g.* reading the board temperature, causing the board front panel lights to flash according to a given pattern, and so on. These tests should be tried if it appears that the LTU is not working properly. More details are given in section 10.

4. Stdfuncs.

4.1 VME r/w. Using this window allows the user to read or write to specific VME registers on the LTU. A full list of the LTU registers is given in Appendix B, and further details of additional functions are given in the *LTU Software Model.* For non-expert users, the use of this window has been largely superseded by the custom options in the rest of the LTU control window system.

4.2 Multiregs A more convenient way of controlling several VME registers at once. ********Add more details* ********.

5. The CTP Emulator

On clicking the **CTP Emulator** button on the main screen, the **CTP Emulator** main screen appears (see figure 2). This is separated into three parts.

5.1 Sequence Selection and Editing. This comprises three buttons.

•**Sequence** gives a drop-down list of configurations already prepared. The list consists of all files with extension `.seq` (ASCII files) in the directory

`$VMECFDIR/CFG/ltu/SLM/`

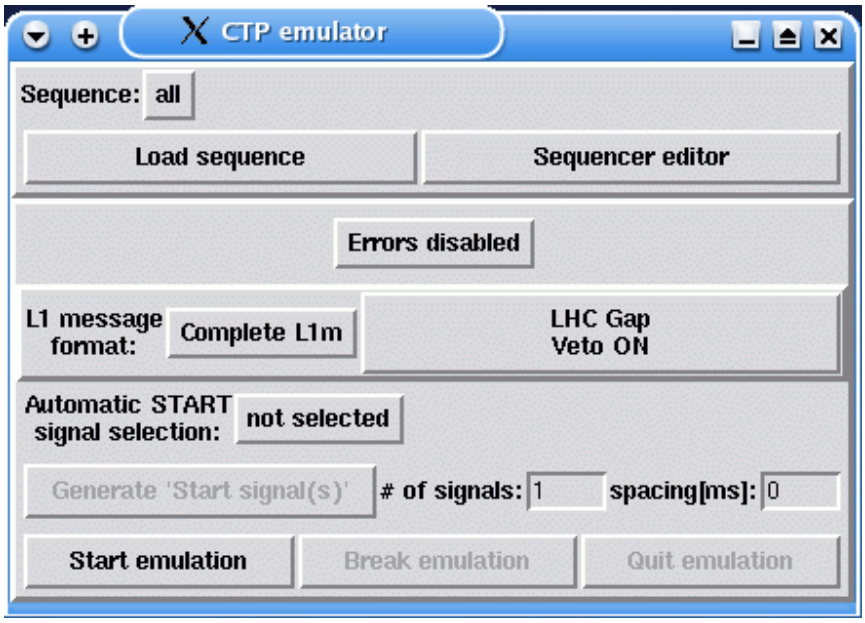


Figure 2. *The main **CTP Emulator** window, in the case where* **Errors disabled** *has been selected.*

Note that if new entries are included in this directory the 'refresh' option should be selected to make them appear. New entries can be added by manual editing or by using the sequencer editor.

•**Sequencer Editor.** Clicking this button opens the sequencer editor window, shown in figure 3. This window is a gui allowing the user to read, edit and write `.seq` files.

If reading/editing an existing file the first step is to choose from the files shown in the **List of .seq files** window. The chosen file name is entered in the **Input sequence file name** entry and **Read sequence** clicked.

The file has two parts. The yellow **Allow Errors with** window defines the default behaviour for error emulation when this sequence file is loaded. Errors are enabled by default in the steps shown as selected. Note that more detailed error control can be achieved using the **CTP Emulator** window, as explained below.

The list of sequences is specified in the lower left hand window, marked **Sequence List**. The list shows the programmed list of sequences. Any sequence on the list can be selected using the radio buttons. For the selected sequence, the list of active classes at Level 1 and Level 2 are shown, together with the detector clusters and user bits. These define the L1 and L2 messages, but *are not otherwise checked for syntax*. To

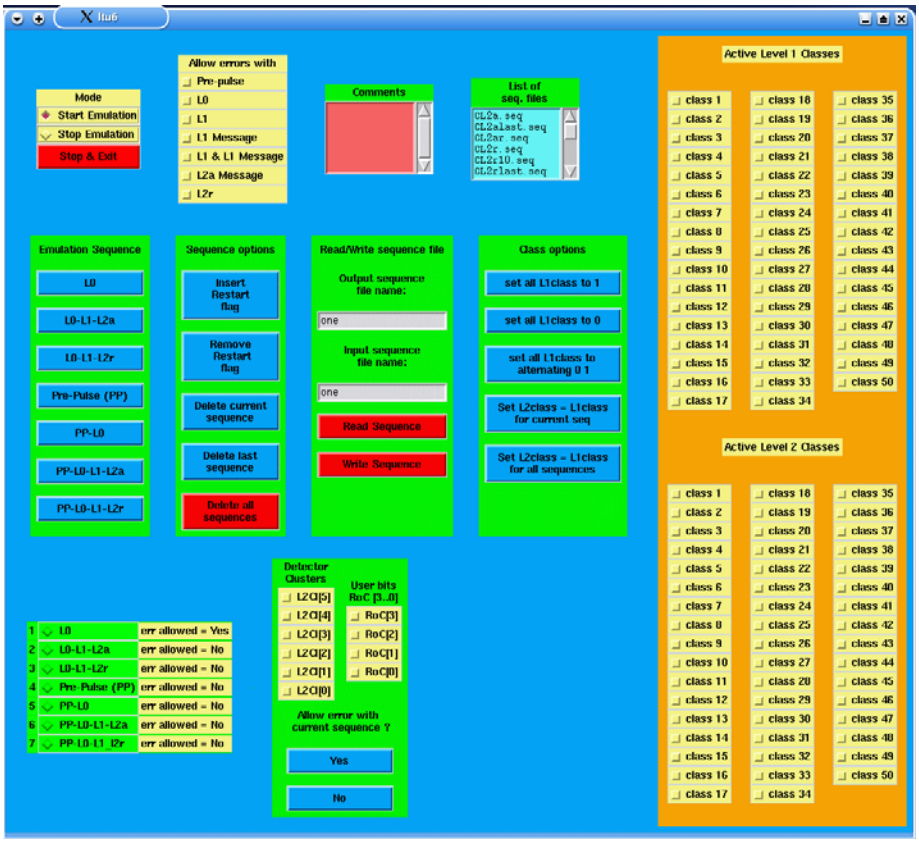


Figure 3. *The* ***Sequencer Editor*** *window.*

facilitate entry, the classes can be filled rapidly using the 'Class Options' selection.

The **sequence options** window includes an option to insert and remove restart flags. Inserting a restart flag defines a loop in a sequence list. If there are more sequences below the restart flag, they can be accessed only after a **break** command, which can be issued from the main **CTP Emulator** window when an emulation is active. It is also possible to delete a sequence or to replace one. If no current sequence is selected, the new sequence is added at the end of the sequence list.

5.2 **Setting of Conditions.** This comprises the three buttons **Errors enabled/disabled**, **L1 Message Format** and **LHC Gap Veto**.

•**Errors disabled/enabled.** For the moment, we consider **errors Disabled**. Selecting **errors enabled** changes the options on the window, and will be discussed below. Note that any error definitions specified in the sequence editor will *not* be activated is this selector is not set to **Errors Enabled**.

•**L1 Message Format.** As described in the *LTU Preliminary Design Document*, the L1 message can appear in full or shortened form (header only) or be suppressed. The selector (drop-down list) selects which of these options will be used.

•**LHC Gap Veto.** The **LHC Gap Veto** toggle switch decides whether the start signal is allowed to be generated for bunch crossings corresponding to the 'large' LHC gap (by default BCs 3446 to 3563). If the veto is ON, (standard state) start signals are not allowed in this interval, if off, all BCs are allowed.

5.3 Emulation Control. The bottom part of the panel defines the conditions for starting and stopping an emulation, and allows emulations to be started, sequence break points to be controlled, and emulations stopped. The buttons that appear depend on the selections made.

•**Automatic START Signal Selection.** This gives a drop-down menu with four different modes.

Pulser input can be *edge-sensitive* or *level senstitive*. With this option selected, the start signal is generated when a signal arrives on the external pulser input. In *edge sensitive* mode, a start signal gives rise to a single pulse at the next clock after the rising edge; in *level sensitive* mode, start signals are generated for each bunch crossing when the START signal is on.

•**Random Start.** Start signals are generated with a random rate. The rate is controlled by the parameter which appears in the new window marked 'rate'. 'Rate' specifies the number of start signals, randomly distributed, which occur in each interval of 2^{31}-1 bunch crossings, i.e. approximately 53s.)

•**BC scaled down.** When this option is selected, a window appears requesting an input for the downscaling factor, *d*. The pulser rate is then $40\times10^6 \big/ (d+1)$.

5.4 Errors Enabled

As explained in section 4.2, the default setting on startup of the **Errors Enabled/disabled** toggle switch is **Errors Disabled**. If **Errors Enabled** us selected, the aspect of the **CTP Emulator** main window changes. The layout is shown in figure 4. As can be seen the functions of the window are essentially the same,

except that a number of new options related to error generation become available. The error buttons are shown on a pink background.

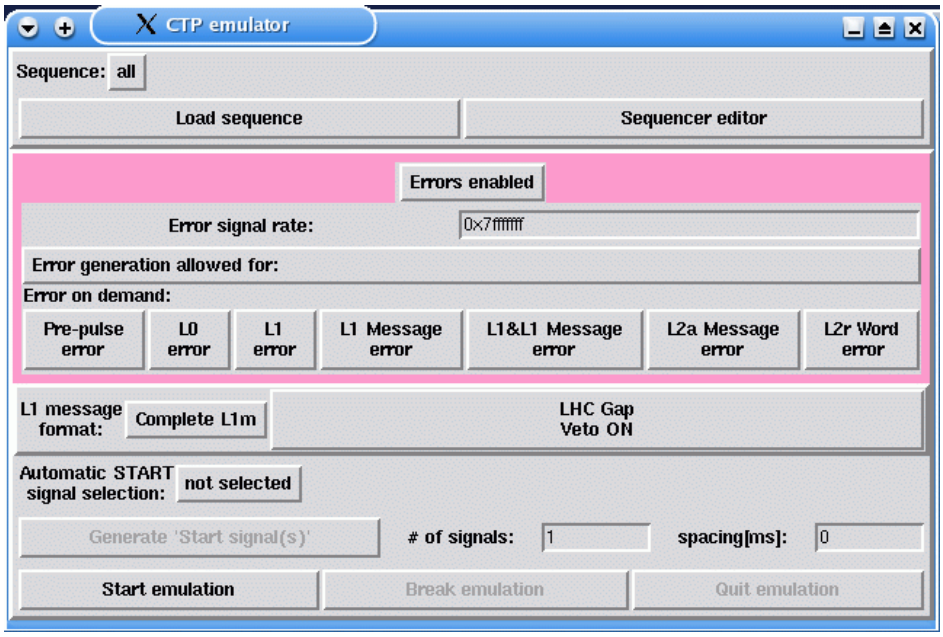


Figure 4. *The **CTP Emulator** main window with errors enabled.*

The generation of errors involves some interplay between the **CTP emulator** main window and the **Sequence editor** window. If a sequence list is selected in which errors have been programmed for certain sequences, which requires flagging **error prone** for the sequence and specifying which steps (PP-L0-L1-L1m-L2a-L2r) can have errors, then selecting **Errors enabled** activates all these choices. An explanation of the error options is given in Table 1. Note also that errors will only be produced if the specified error prone stage in trigger generation occurs in the given sequence. For example, if a sequence **L0** is chosen, and errors are flagged for **L1 message**, no errors will be produced since the selected stage for producing errors is not applicable.

If a sequence is loaded in which error generation is specified and errors are enabled, the **Error Generation allowed for** button will indicate the types of error allowed. When the button is pushed, a drop-down menu appears indicating which types of error have been selected. If an error type is not selected, or is manually unselected, this type of error will not occur. If a new

type of error is selected, it will occur in all error prone sequences where it is applicable.

Table 1. *Explanation of the error conditions for* error prone *operation.*

Name	**Explanation**
Prepulse	Prepulse is missing.
L0	L0 is missing.
L1	L1 pulse is missing.
L1message	L1 message is missing
L1 & L1 message	Both L1 and L1 message are missing.[2]
L2a	L2a message is missing.
L2r	L2r flag is missing.

There are two options for error generation, **Random Error** and **Error on Demand**. The **Random Error** option generates errors randomly, according to the value set in the Error Signal rate register (ESRR). The value set in ESRR is compared with a 32 bit random number, generated for each bunch crossing. If the generated random number is lower than the ESRR value and it coincides with the *Error-prone* signal, this signal/message is suppressed.

From this, setting the ESRR to `0x7fffffff` leads to error generation for *all* the chosen error-prone signals/messages, while setting 0 means disabling errors.

The **Error on Demand** option is controlled by the 7 buttons placed below the 'Error on Demand' label. Pressing an one of these buttons asserts the corresponding request flag, which is applied at the next possible occasion, i.e. when there is next a signal/message susceptible to the chosen error type.

Note the difference between these two types of error generation: while a random error appears only if there is a susceptible signal or message in the chosen bunch crossing, *error on demand* acts on the next available bunch crossing with a suitable error-prone signal/message. 'Errors on demand' can be requested in parallel, while random errors are generated.

[2] Note that, when this option is chosen, it overrides the **L1** and **L1** message options, *i.e.* it is not possible to obtain errors in L1 or L1 message alone if L1&L1 message is set, even if either one of L1 or L1 message is also set.

6. Scope Signals

The LTU has two outputs offering signals for an oscilloscope, labeled **A** and **B**. Copies of signals received as inputs or generated on the LTU board can be selected to appear on these inputs, allowing checks of relative timing. On selecting the **Scope signals** option on the main panel (see figure 1), a panel appears indicating the current scope selections (see figure 5). Clicking on either of the boxes with the currently selected signals reveals a tear-off panel with a list of all the available signals for the given scope output, with the currently selected one indicated. Use the mouse to change the selection. The tear-off panels corresponding to figure 5 are shown in figure 6. Note that the relative timing of the signals is correct to about 1-2 ns, but depends on the internal path of the signals to the scope outputs.

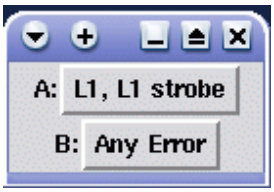
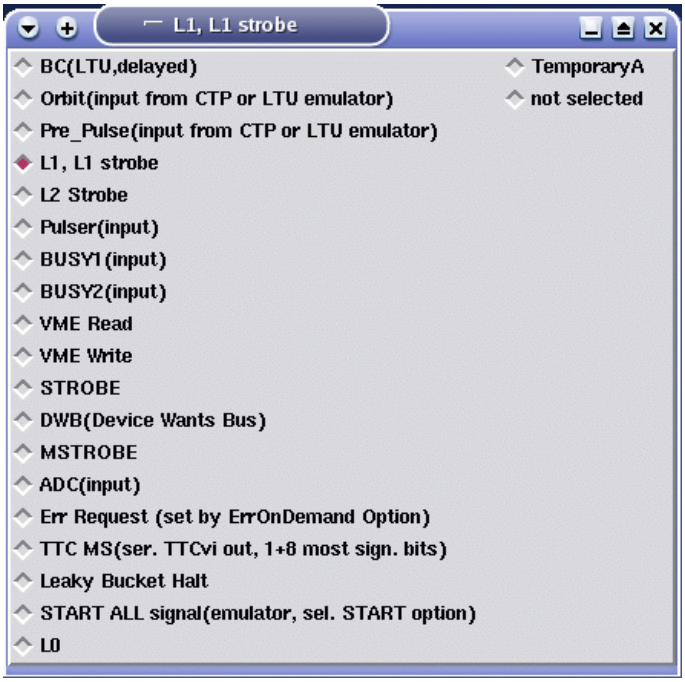


Figure 5. *Main **Scope signals** panel, showing the currently selected signals for outputs **A** and **B**.*

(a)



(b)

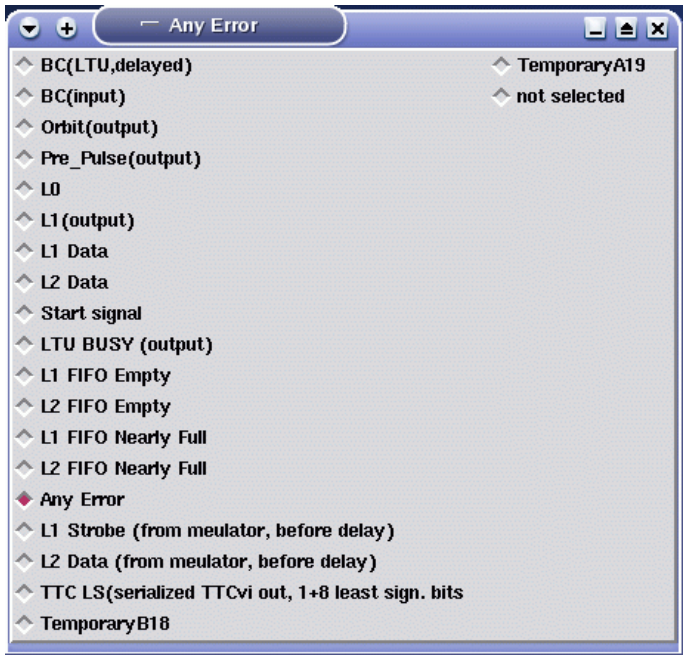
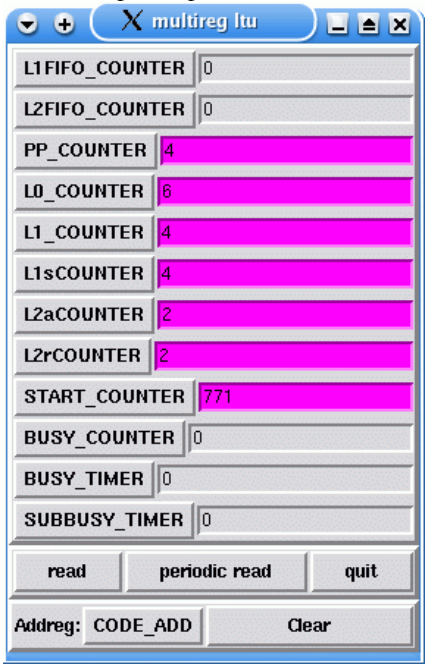


Figure 6. (a) *Menu of available signals for oscilloscope output **A;*** (b) *same for oscilloscope output **B**.*



7. Counters

The counters option on the main menu is used to access the scalers on the LTU. Selecting this option brings up the counters panel, shown in figure 7.

Figure 7. *Menu of counters, showing an array of selected counters, the* ***read****,* ***periodic read****, clear and* ***addreg*** *options. In this example the counters are displayed in decimal format. Scalers in purple have changed since the last* ***read****.*

The list of counters displayed is programmable. Selecting the **Addreg** button produces a drop-down list of counters, each of which can be selected and thus made to appear on the panel. All the counters that appear on the panel are read (once) on pressing the **read** button, or regularly every few seconds on pressing the **periodic read** button. The counters are all reset together on pressing the **clear** button. Note that the display of the content of the scaler can be in either hexadecimal or decimal format. On startup the default is hexadecimal, but right-clicking a counter will cause it to toggle between hexadecimal and decimal.

8. Snapshot Memory.

For diagnostic purposes, the LTU has a *snapshot memory*, which allows all the activity of the LTU top be monitored at 25 ns intervals for a period 26.6ms (i.e. 294 full LHC orbits). The snapshot memory can be controlled using the **snapshot** option on the main menu. Clicking this option brings up the snapshot window, which is shown in figure 8.

Figure 8. *The snapshot window. The options include selecting whether the snapshot should cover the period before or after the snapshot signal. If **after,** then the **start** signal has to be specified. The **dump** and **show** buttons are used for selecting options for the display of the snapshot contents.*

The snapshot facility operates in two modes, depending on whether it is timed by a **start** or a **stop** signal. In **Start after** mode, the snapshot runs for the specified period as from when the **start** signal is received. There are several options as to which signal is used to specify the **start**, and the selection is made using the drop-down menu in the **Scheduled start** box. In the **Start Before** mode, the snapshot memory is being continuously filled and overwritten, but is frozen when the specified signal is received. Again, the snapshot can be stopped using a variety of signals as specified in the menu in the **Scheduled start** box. A full list of possible start signals is given in Table 2.

Table 2. *List of possible start signals for an LTU snapshot, as specified in the **Scheduled Start** menu.*

An LTU snapshot consists of a possible 18 data bits per bunch crossing. The data bits are given in Table 3. The results are written into a file `SSM.dump` in the standard `$VMECFDIR/WORK` directory in binary form. This file is then available for offline analysis.

Table 3. *Bit definitions for the data word in an LTU snapshot*

.

Bit	Meaning	Abbreviation
0	Orbit	**ORBIT**
1	Pre-pulse	**PP**
2	L0	**L0**
3	L1/L1 Strobe	**L1S**
4	L1 Data	**L1DATA**
5	L2 Strobe	**L2S**
6	L2 Data	**L2DATA**
7	Sub-detector BUSY (synch.)	**SBUSY**
8	LTU BUSY (synch.)	**ALLBUSY**
9	L1 FIFO Nearly Full	**L1NF**
10	L2 FIFO Nearly Full	**L2NF**
11	Leaky Bucket Halt	**LBHALT**
12	MSTROBE	**VMEM**
13	TTC LS [3]	**TLS**
14	TTC MS[4]	**TMS**
15	STROBE	**VMES**
16	START ALL signal (emulator)	**ALLSTART**
17	*Any Error*	**ANYERR**

[3] Serialized TTCvi output: a header set to 1 and 8 LS bits.

[4] Serialized TTCvi output: a header set to 1 and 8 MS bits.

In order to facilitate analysis of the snapshot, it is also possible to view the snapshot in a partially pre-processed form. Once a snapshot file has been created, pushing the **Show** button will start the interpretation step. In this case, a file `SSMa.txt` is created in ASCII format, which contains an interpreted version of the `SSM.dump` file. The file is opened in the **logfile** window. Only non-empty bunch crossings (i.e. those in which at least one of the 18 data bits in the SSM data word is non-zero) are displayed, with the event indicated by the appropriate SSM data abbreviation, as listed in table 3. An interpreted example of a portion of a snapshot file is given in figure 9.

The main points of the `SSMa.txt` file are listed below.

(i) Timings are given in bunch crossings taken from the beginning of the snapshot.

(ii) The format for the signals is to give the signal abbreviation (see Table 3), then, after a '/', the length of the signal in bunch crossings. Thus for example, "0: ORBIT/40" means an orbit reset signal was initiated in bunch crossing 0 and lasted 40 bunch crossings.

(iii) At the beginning of a transmission of L1 or L2 data through VME, a log of the serial transmission of data is given (bit 4 of snapshot summary for L1 data, bit 6 for L2 data). This takes the form of a listing of the data words (which take 5 and 8 transmissions for L1 and L2 words respectively), followed by a summary of the activated trigger classes, given as a string of $base_4$ numbers.

(iv) For each of the data transmissions, the transmission of each word is registered as a sequence (VMEM-TTCMS-TTCLS). The first hexadecimal digit is a flag indicating the message type (1 = L1 Header, 2 = L1 data, 3 = L2 header, 4 = L2 data) and the next two hexadecimal digits give the data, which ties up with the L1/L2 data summary printed out at the beginning of the transmission.

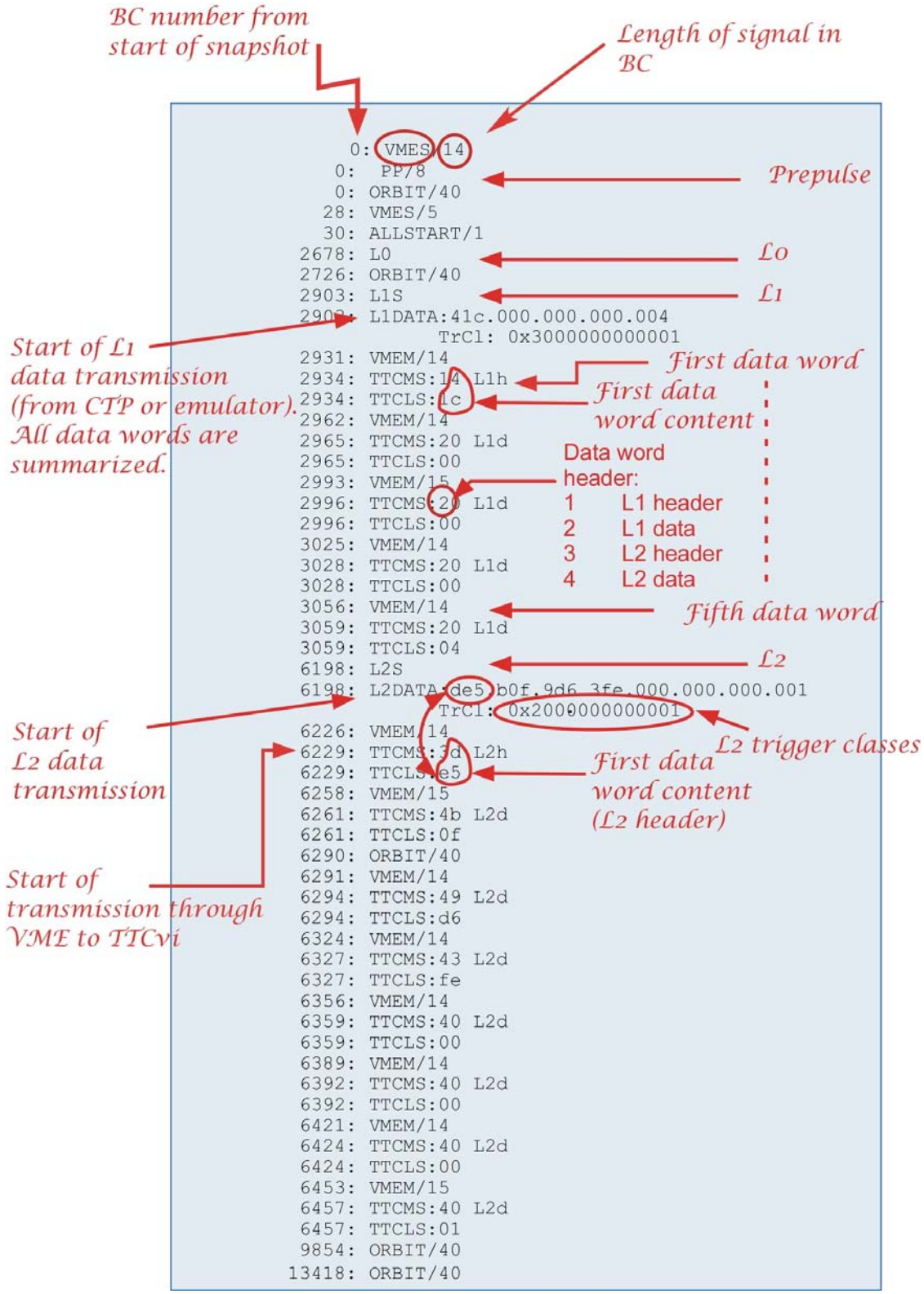


Figure 9. *An interpreted snapshot file of the kind* `SSMa.txt`. *Note the annotations. Only non-empty bunch-crossings are listed.*

9. Configuration

The Configuration menu, shown in figure 10, offers several possibilities. The most important are the top two options, which determine whether the LTU is to be in GLOBAL or STANDALONE mode. In addition there are three other configuration options, each of which activates a new window. (In figure 10, two of those options have been selected, and the associated windows opened, which is why the options are shown in grey.) The other options are described below.

Figure 10. *Main configuration window.*

- **setBUSY option.** Selecting the **setBUSY** option opens a further window, as shown in figure 11. The purpose of the window is to enter a value which specifies which BUSY inputs and outputs are activated. The code value is computed as shown in Table 4. By default, BUSY inputs are disabled.

Figure 11. setBUSY *window. Inputs are enabled or disabled, according to the value entered. See Table 4 for possible values.*

Table 4. *Possible values of* binputs.

binputs	BUSY1	BUSY2
0	disabled	disabled
1	enabled	disabled
2	disabled	enabled
3	enabled	enabled

- **setTTCint option.** Selecting the **setTTCint** option allows one to adjust a few of the TTC interface options on the LTU. A single number is entered in the window, shown in figure 12, whose bits are made up according to the values in Table 5.

Figure 12. setTTCint *window. See Table 5 for an explanation of the input values.*

Table 5. *Assignment of bits for the* setTTCint *input word.*

Bit	Explanation	FLAG=0	FLAG=1
0	Delay **Orbit** by 12.5 ns	No delay	Apply delay
1	Delay **L1** by 12.5 ns	No delay	Apply delay
2	L1 Polarity (TTCex +;TTCvi -)	+ve (default)	-ve

- **ADC_Scan.** The ADC_Scan window is used to check and to set the synchronization between the CTP clock and the local LTU clock. This procedure cannot be applied when the LTU is running in standalone mode with no input on the CTP connector. The window is shown in figure 13. When a scan is performed, the number of cycles to be used has to be specified (it can be quite large, as, for example 10^6 cycles takes less than one second to measure and analyse). A delay scan plot is displayed and a preferred value is suggested. The user can either accept the suggested value or input a new one.

Figure 13. Delay scan window after a delay scan has been performed. The suggested value corresponds to the maximal excursion of the measured ADC value, corresponding to the overlap between the two clocks to be synchronized. The details of this measurement are explained in the LTU preliminary design document, section **3.14.12**.

10. Simple Tests

The **Simple Tests** window on the main panel offers a further panel with a number of simple tests and procedures. The current set of tests is shown in figure 14, and is described below.

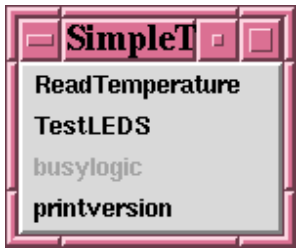


Figure 14. *The main **Simple Tests** panel. More options may be added later.*

- **ReadTemperature.** When this options is selected, the board temperature in degrees Centigrade is returned in the **log** window.

- **TestLEDS.** With this test, the LEDs on the front panel are set flashing in a cyclic pattern.

- **busylogic.** When this option is selected, the panel shown in figure 15 appears. It allows certain tests of the BUSY logic to be performed. When the **start** button is activated, the BUSY counter and both BUSY timers are cleared, prior to performing the requested number of BUSY cycles (see below). At the end the timers and counters can be checked for consistency.

Figure 15. busylogic window.

There are three parameters to be set. The **source** parameter has three bits, and indicates where the BUSY input is coming from (0 = ENABLE BUSY1; 1=ENABLE BUSY2; 2 = SOFTWARE BUSY.) More than one source can be selected, so that, for example, "7" means *all* inputs are selected. The **busys** parameter indicates how many cycles should be executed in the test. The **length** parameter indicates the length of the BUSY signal in units of 3.8 µs, so that for example "4" means a BUSY length of 4×3.8 = 15.2 µs. The timers and counters are selected as described in section 6 above.

- **printversion.** When the **printversion** button is pressed, a number of parameters about the status of the LTU and the software used for it are printed in the log window. These are:

`CODE_ADD`	`0x56` for an LTU board.;
`SERIAL_NUMBER`	the serial number of the board;
`VERSION_ADD`	the VME FPGA firmware version;

`LTUVERSION_add`	the LTU FPGA firmware version;
`BC_STATUS`	the Bunch Crossing status (should be `0x2`, indicating PLL locked, BC OK).

PART 2. EXAMPLES

Example 1. Find the serial number for the LTU

There is more than one way to find out the serial number of the LTU. When the **log** window is opened, the information available through the **printversion** command, which includes the serial number of the LTU, is printed. If for some reason this is missed, the information can be obtained again by running **Simpletests** → **Printversion**. The resulting information in the **log** window is shown in figure 16.

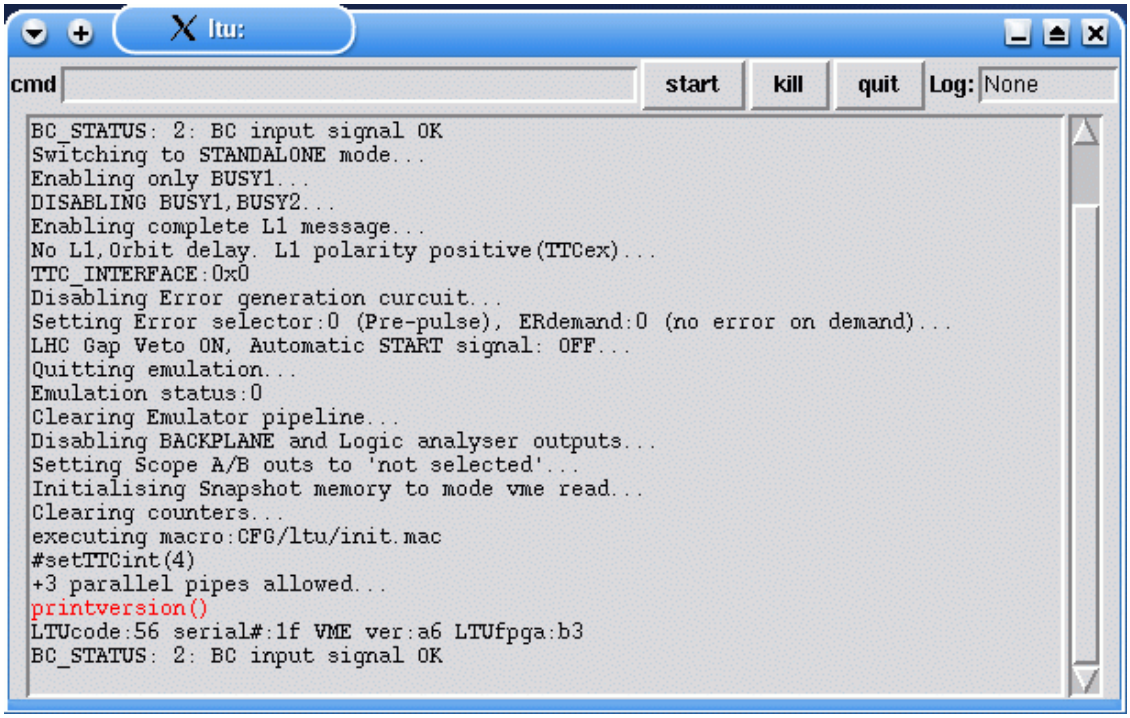


Figure 16. Screen shot of the **log** window after running the **printversion** command. In this case the serial number is `1f`, i.e. 31.

Example 2. Configure the LTU to work with a TTCvi

The default layout for the LTU is to send its L1 output directly to the TTCex, in which case the output polarity should be *positive*. If instead the LTU is to be fed to a TTCvi, then the output polarity should be *negative*. In order to select this option interactively, go to the **configuration** menu, and select **Configuration → setTTCint → 4 → start**. (See section 9 for an explanation.) The resulting log file is shown in figure 17.

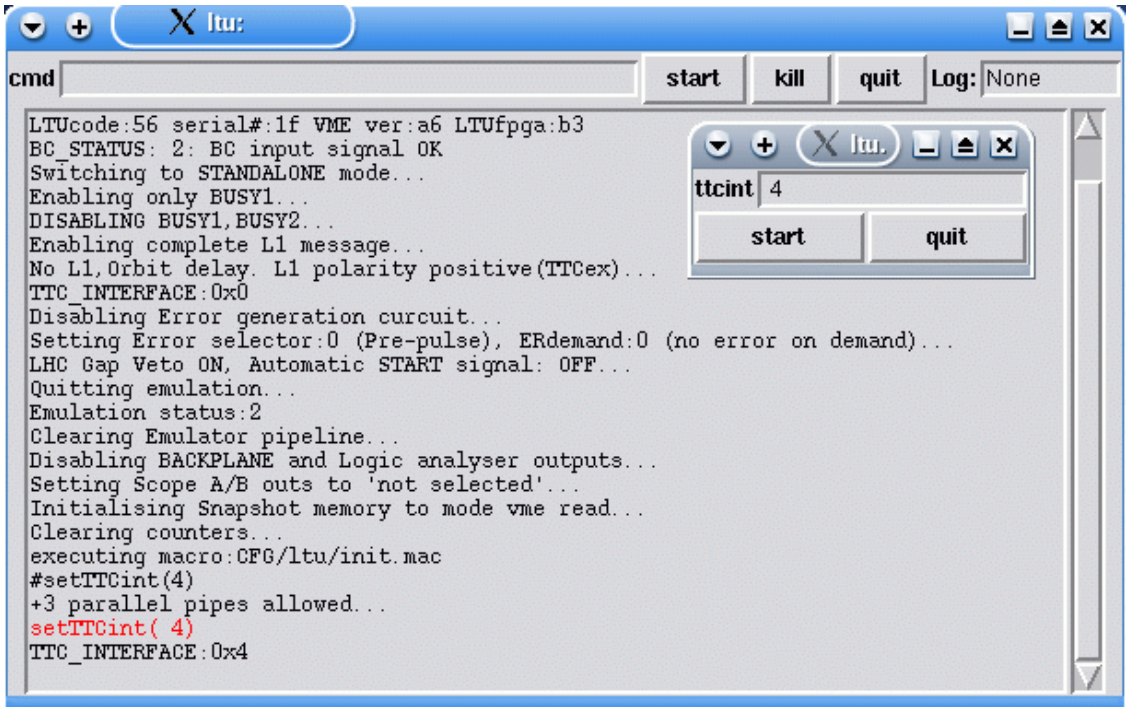


Figure 17. Screen shot of the two relevant windows for adjusting the TTC interface settings. The **setTTCint** window shows the options selected, and the log window records the action taken when the **start** button was pushed.

Example 3. Start an emulation sequence with successful L2 triggers

First select the **CTP Emulator** window, and choose the **L2a** sequence. Load it by pressing the **load sequence** button. The sequences loaded can be seen in the **log** window. Each line consists of the mnemonic code for the sequence and the actual bit code for the sequence. Next select the **start** signal source. For example, a scaled-down BC can be chosen. For this case select **BC** and a downscaling factor of 40 000, which will produce **start** signals with a frequency of 1 kHz. Once these steps are completed, pressing **start emulation** will start an endless sequence of **L2a** triggers (because of the restart flag in the L2a sequence). The sequence can be stopped either by choosing **not selected** for the **automatic START signal selection** option, or by pressing the **Quit emulation** button. The relevant windows are shown in figure 18.

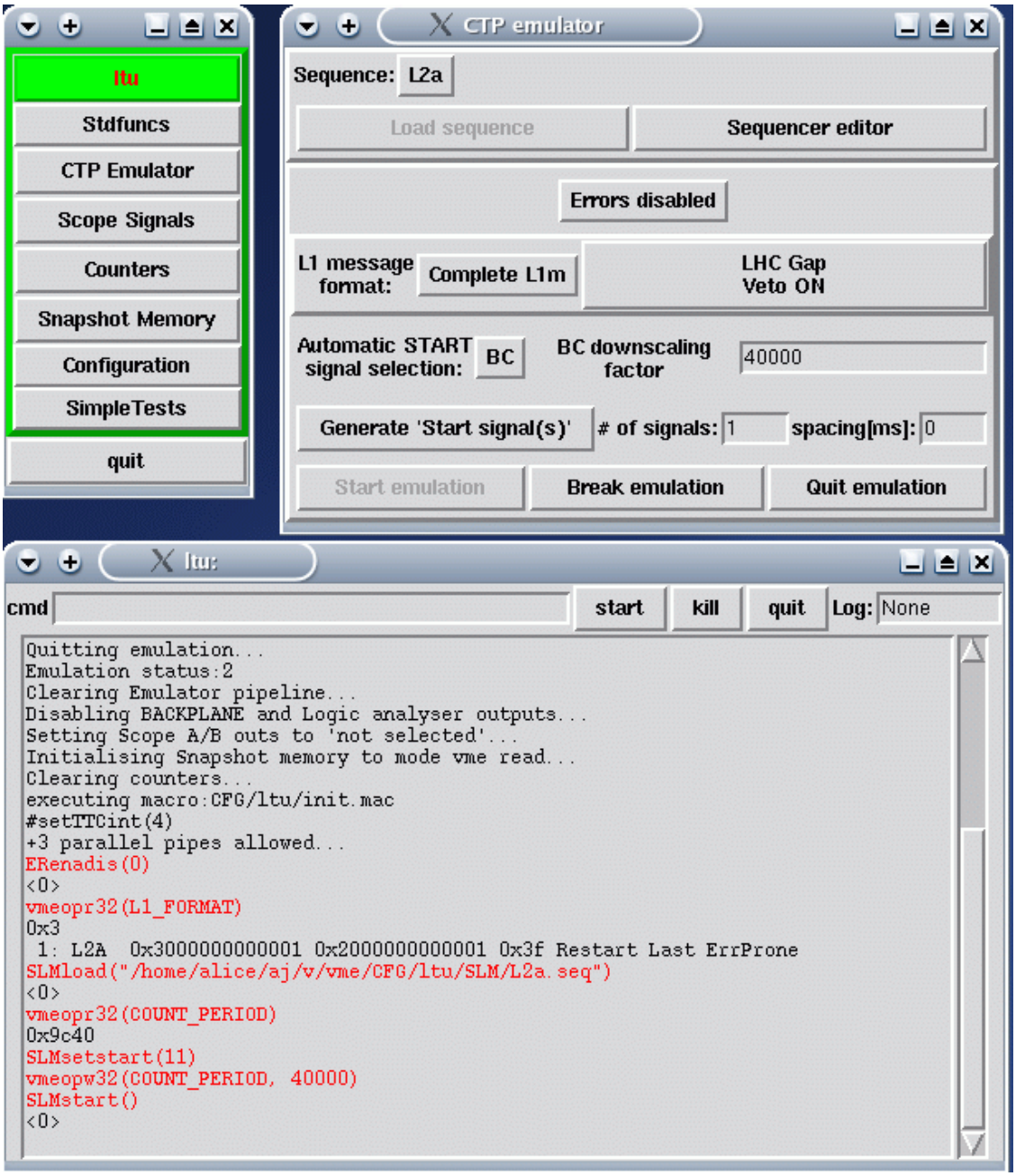


Figure 18. Relevant windows for the setting up of the L2a sequence. The main **CTP Emulator** window is shown with the appropriate selections already made. The **log** window indicates the selections made. “Complete L1m” is the default option for the L1 messages, and the content of the L2a sequence is shown in the **log** window. Note the “Restart” which leads to the repeating sequence. In this case “error prone” has been selected, which means that if the sequence is run with **errors enabled**, the occasional error buttons are active.

Example 4. Select signals for scope output.

Select the **Scope signals** option on the main LTU window, and then use the drop-down menus to select the required signals for scope outputs A and B. In figure 19, the L1 strobe and L0 signals have been chosen.

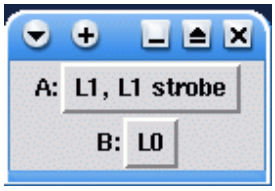


Figure 19. Scope signals window. Clicking either of the scope "A" or scope "B" windows produces a drop-down menu of signals, and in this case **L1 strobe** and **L0** have been chosen.

Example 5. Monitor scalers for a running LTU sequence.

Let us assume that a procedure such as that in example 3 has been followed to start an emulation sequence. It is now possible to monitor scalers for this sequence. Select **Counters** in the main LTU window. The list of scalers can be modified using the **Addreg** drop-down list, as explained in section 7. The most convenient mode of operation is to select **Perodic read**, in which case the window is updated roughly every second. Single reads are also possible. The relevant windows are shown in figure 20.

Example 5. Make a snapshot of the current process.

The order in which a snapshot is set up is important. First prepare a snapshot. Open the **Snapshot memory** window and select **Start After when StartSignal Selected** in the **Scheduled start** window. Next open the **CTP Emulator** window, select a sequence (see example 3) and start the emulation (press **Start emulation)**. (N.B. Do not at this point make a selection in the **Automatic START signal selection** box. We now have an active emulation, but no triggers produced as there is as yet no start signal. At this point if either (*a*) a choice is made in the **Automatic START signal selection** box or (*b*) the **Generate Start signal** button is pushed in the **CTP Emulator** window, the snapshot will start. The actions are all listed in the **log** window, as shown in figure 21. The result is that a file `SSM.dmp` is created, as indicated in the **log** window. Finally, press **Show** in the **Snapshot Memory** window to process the file `SSM.dmp` to make a file `SSma.txt`, containing an interpreted file, as explained in section 8. A new window opens to show the file `SSMa.txt`, as shown in figure 22.

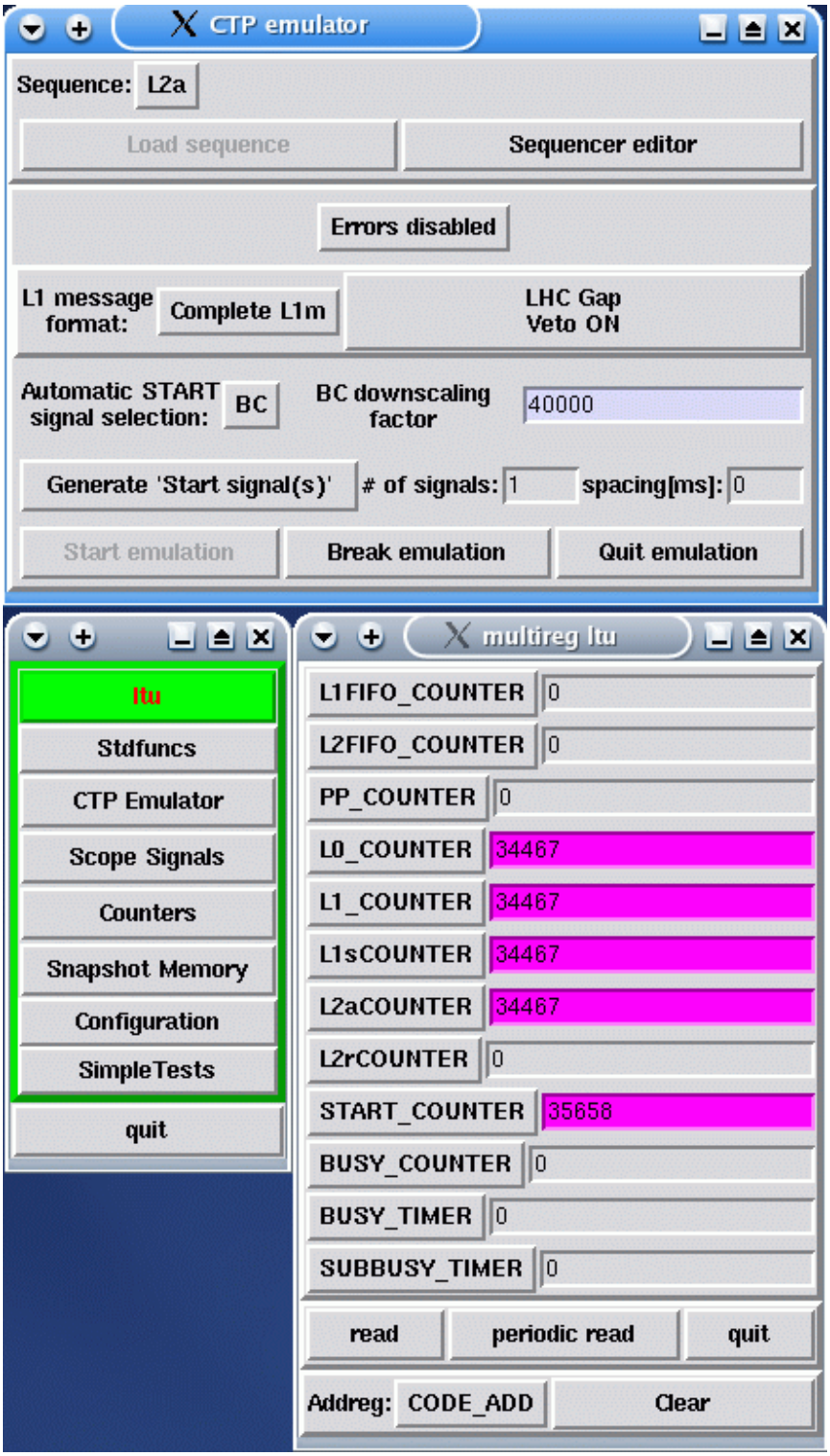


Figure 20. With the **CTP Emulator** window set up as for example 3, push the **Counters** button on the main LTU window. The **Counters** window appears. Note that the items in purple have changed value since the last read.

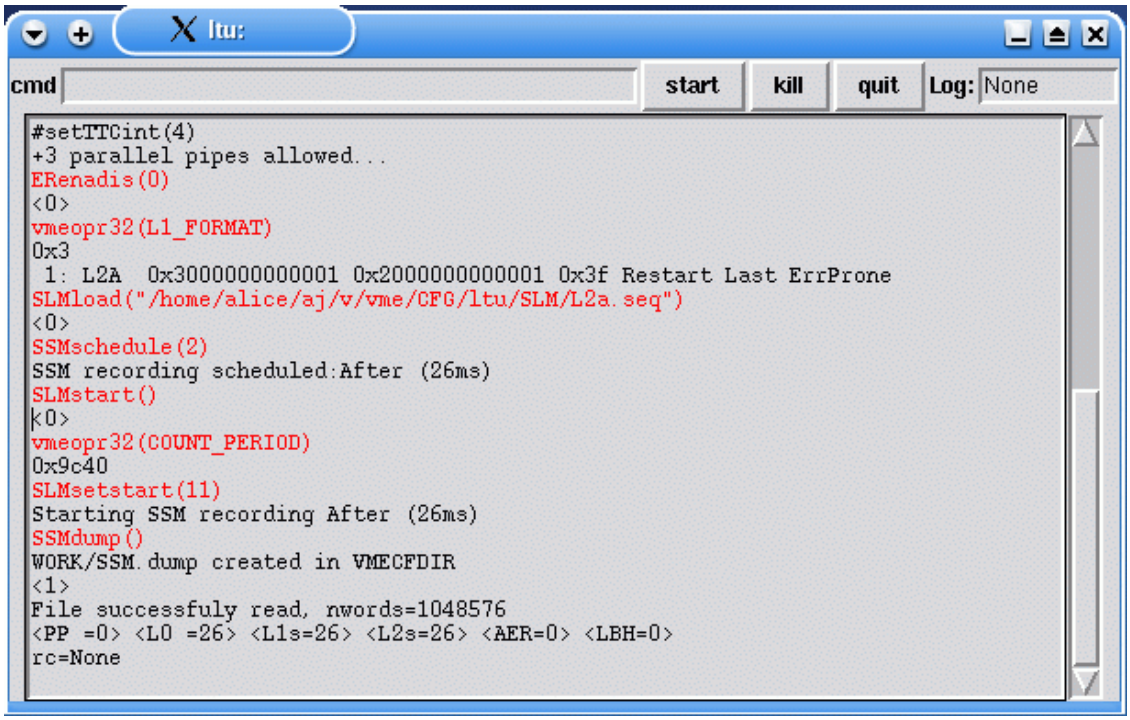


Figure 21. Log file for a snapshot. Most of the steps immediately before the snapshot is made in **Start after** mode are done in the **CTP Emulator** window, as indicated in the figure. Once the emulation starts, the snapshot file `SSM.dmp` is created.

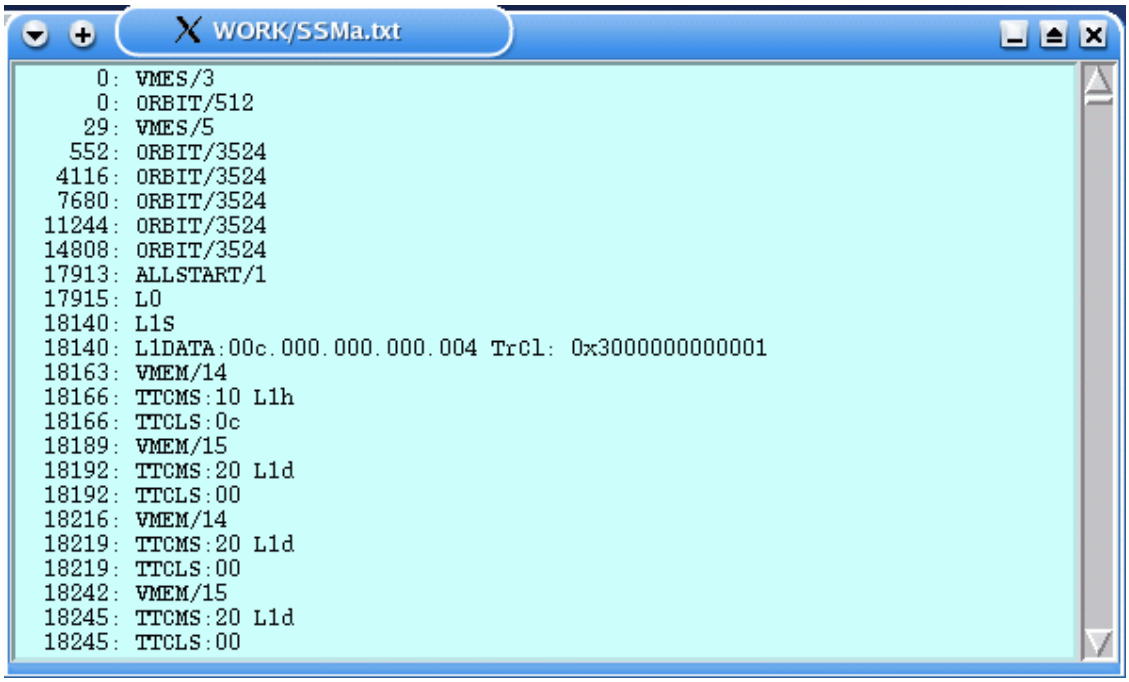


Figure 22. New window created by the **show** command in the **Snapshot Memory** window, in which the interpreted file `SSMa.txt` is displayed. See section 8 for an explanation of how to read the file.

APPENDIX A. Installation of the LTU Software.

In order to install the LTU software on a Linux platform, first create a directory (let us call it v), and unpack the `tar` file in it:

```
cd v
tar -xvf../vmebltu.tar
```

Then edit the following files:

`v/scripts/vmebse.bash`
 	
`VMEDRIVER -VMECCT`
`VMERCC`
`AIX`
`VMECFDIR= ...v/vme -`
`VMEBDIR= ...v/vmeb`
	Setting for 3 environment variables
	(CCT vme driver)
	(Atlas vme driver)
	(Motorola SBC with AIX)
	specific software (ttcvi, ltu,...)
	common software (for any vme board)

`v/vmeb/vmeai.$VMEDRIVER`	**link and include directories for VME driver**
-	
v/vme/ltu/ltu.c	**In first line, modify BASE address of the LTU board. Base address of the LTU board is 0x81*x*000, where *x* is set by the switch on the LTU board. The TTCvi base address has to be set to 0x80*x*000 (there are 4 switches on TTCvi board)**
v/vme/ttcvi/ttcvi.c	On the first line, modify the base address of the TTCvi board

Start

Run

`v/scripts/vmebse.bash`	**sets VMEDRIVER, VMEBDIR, VMECFDIR environment variables and some useful aliases convenient in .bash profile)**
vmecomp ltu ttcvi	**Compiles ltu, ttcvi software**

cd v/vme/SSMANA

`gcc ssmana.c -o ssmana.exe-`	**compiles Snapshot memoryanalysis program**
`v/scripts/ttcvi`	**Initialize ttcvi board (only once after power-up)**
`vmecrate ltu`	**Start ltu software**

Wiring of the TTC partition

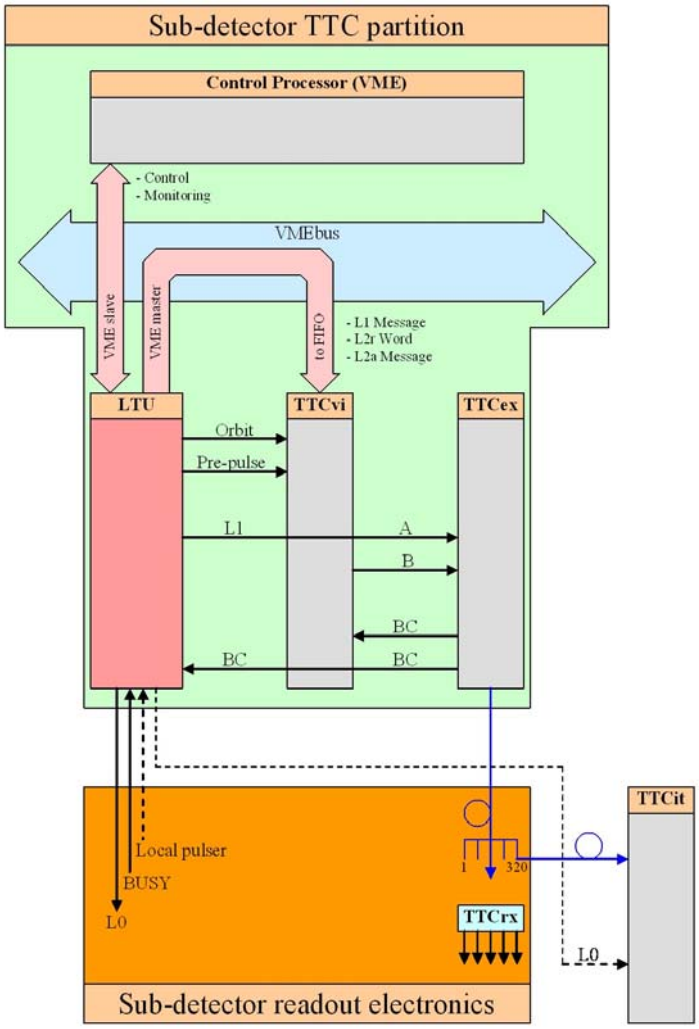


Figure 23. Layout and connections for the LTU when not connected to the CTP.

For more details see chapters 3.5 *LTU connections*, and 3.14.2 *Signal connections in the TTC partition* in the LTU PDR document and the TTCvi/TTCex manuals.

APPENDIX B. VME Registers.

The full list of VME registers for the LTU is given in table 2 below.

Table 2. List of VME Registers and addresses for the LTU.

Symbolic name	Address	Explanation	Reference
CODE_ADD	4	Board type (0x56 for LTU)	
SERIAL_NUMBER	8	unique serial number of the board	
VERSION_ADD	C	VME FPGA firmware version	
SOFT_RESET	28		
TEMP_START	58	LTU temperature	temperature
TEMP_STATUS	5c	LTU temperature	temperature
TEMP_READ	60	LTU temperature	temperature
LTU_VERSION_ADD	80	LTU FPGA firmware version	
BLINK	C0		
BC_STATUS	C4		
BC_DELAY_ADD	C8		
ADC_START	CC	Automatic alignment	ADCreadout
ADC_DATA	D0		ADCreadout
ERROR_ENABLE	D4	Error emulation	ERRemulation
ERROR_STATUS	D8		ERRemulation
ERROR_SELECTOR	DC		ERRemulation
ERROR_DEMAND	E0		ERRemulation
ERROR_RATE	E4		ERRemulation
ORBIT_CLEAR	E8	Clear emulator's orbit counter	
EMULATION_START	EC	Start emulation	Emulator
QUIT_SET	F0		Emulator
BREAK_SET	F4		Emulator
LAST_BC	F8	Default: 3563	Constants
PREPULSE_BC	FC	Default: 3436	
CALIBRATION_BC	100	Default: 3556	
GAP_BC	104	Default: 3446	
ORBIT_BC	108	Default: 1	
L1_DELAY	10C	Default: 224	
L2_DELAY	110	Default: 3520	
SLM_ADD_CLEAR	114	Sequencer memory	Emulator
SLM_DATA	118		
EMU_STATUS	11C		
START_SET	120		
SOFT_TRIGGER	124		
RANDOM_NUMBER	128		
COUNT_PERIOD	12C		
TTC_INTERFACE	130	L1 polarity, Orbit delay setting	TTCinterface

STDALONE_MODE	134	GLOBAL:0 STDALONE:1	
BUSY_ENABLE	138	Write 0x3 ->enables BUSY1,2	Busy
BUSY_STATUS	13C		
SW_BUSY	140		
L1_FORMAT	144	bit0=1:L1enable, bit1=0: L1h only	
L1MAX_CLEAR	148		Fifo
L2MAX_CLEAR	14C		
SCOPE_SELECT	150	groupB: 0x1e0 groupA: 0x1f	Scope
BACKPLANE_EN	154		Backplane
FIFO_MAX	158	Max. FIFO occupancy	Fifo
SOFTWARE_LED	15C	dummy write blinks spare LED	
COUNTER_CLEAR	160		Counters
L1FIFO_COUNTER	164		
L1FIFO_COUNTER	168		
PP_COUNTER	16C		
L0_COUNTER	170		
L1_COUNTER	174		
L1sCOUNTER	178		
L2aCOUNTER	17C		
L2rCOUNTER	180		
START_COUNTER	184		
BUSY_COUNTER	188		
BUSY_TIMER	18C		
SUBBUSY_TIMER	190		
COUNTER_MASK	194		
PIPELINE_CLEAR	198		Emulator
SSMcommand	19C	Snapshot memory	Snapshot
SSMstart	1A0		
SSMstop	1A4		
SSMaddress	1A8		
SSMdata	1AC		
SSMstatus	1B0		
TIMING_TEST	1B8	wr only, bit0: 0->disable, 1->enable	

APPENDIX C Adjustment of rates for Error Generation.